Package 'shapley'

May 9, 2026

Type Package

Title Weighted Mean SHAP and CI for Robust Feature Assessment in ML Grid

Version 0.7.0

Depends R (>= 3.5.0)

Description

This R package introduces Weighted Mean SHapley Additive exPlanations (WMSHAP), an innovative method for calculating SHAP values for a grid of fine-tuned base-learner machine learning models as well as stacked ensembles, a method not previously available due to the common reliance on single best-performing models. By integrating the weighted mean SHAP values from individual base-learners comprising the ensemble or individual base-learners in a tuning grid search, the package weights SHAP contributions according to each model's performance, assessed by multiple either R squared (for both regression and classification models). alternatively, this software also offers weighting SHAP values based on the area under the precision-recall curve (AUCPR), the area under the curve (AUC), and F2 measures for binary classifiers. It further extends this framework to implement weighted confidence intervals for weighted mean SHAP values, offering a more comprehensive and robust feature importance evaluation over a grid of machine learning models, instead of solely computing SHAP values for the best model. This methodology is particularly beneficial for addressing the severe class imbalance (class rarity) problem by providing a transparent, generalized measure of feature importance that mitigates the risk of reporting SHAP values for an overfitted or biased model and maintains robustness under severe class imbalance, where there is no universal criteria of identifying the absolute best model. Furthermore, the package implements hypothesis testing to ascertain the statistical significance of SHAP values for individual features, as well as comparative significance testing of SHAP contributions between features. Additionally, it tackles a critical gap in feature selection literature by presenting criteria for the automatic feature selection of the most important features across a grid of models or stacked ensembles, eliminating the need for arbitrary determination of the number of top features to be extracted. This utility is invaluable for researchers analyzing feature significance, particularly within severely imbalanced outcomes where conventional methods fall short. Moreover, it is also expected to report democratic feature importance across a grid of models, resulting in a more comprehensive and generalizable feature selection. The package further implements a novel method for visualizing SHAP values both at subject level and feature level as well as a plot for feature selection based on the weighted mean SHAP ratios.

License MIT + file LICENSE

Encoding UTF-8

Imports ggplot2 (>= 3.4.2), h2o (>= 3.34.0.0), curl (>= 4.3.0), pander (>= 0.6.5)

RoxygenNote 7.3.3

URL https://github.com/haghish/shapley

BugReports https://github.com/haghish/shapley/issues

NeedsCompilation no

Author E. F. Haghish [aut, cre, cph]

Maintainer E. F. Haghish <haghish@hotmail.com>

Repository CRAN

Date/Publication 2026-03-04 06:40:02 UTC

Contents

`feature.selection` *Select top features by weighted mean SHAP (WMSHAP)*

Description

Selects a subset of features from a shapley object. Features can be selected by: (1) specified ‘features‘, (2) ‘top_n_features‘, or (3) WMSHAP cutoff for "mean" or "lowerCI".

Usage

```
feature.selection(
  shapley,
  method = "lowerCI",
  cutoff = 0,
  top_n_features = NULL,
  features = NULL
)
```

Arguments

shapley	shapley object
method	Character. Specifies statistic used for thresholding. Either "mean" or "lowerCI" (default) should be specified. Ignored if 'top_n_features' is provided.
cutoff	Numeric. Cutoff for thresholding on 'method'. Default is zero, which means that all features with lower WMSHAP CI above zero will be selected.
top_n_features	Integer. If provided, selects the top N features by 'mean', overriding 'method' and 'cutoff'.
features	Character vector of features to keep. If provided, it is applied before 'top_n_features'/'cutoff' selection (i.e., selection happens within this set).

Value

A list with:

shapley The updated shapley object.

features Character vector of selected features, ordered by decreasing mean SHAP.

mean Numeric vector of mean SHAP values aligned with 'features'.

Author(s)

E. F. Haghish

---

feature.test *Weighted permutation test for paired difference of means*

Description

Performs a weighted permutation test for the null hypothesis that the weighted mean of (var1 - var2) is zero.

Usage

```
feature.test(var1, var2, weights, n = 2000)
```

Arguments

var1	A numeric vector.
var2	A numeric vector of the same length as var1.
weights	A numeric vector of non-negative weights of the same length as var1 and var2.
n	Integer. Number of permutations (default 2000).

Value

A list with:

mean_wmshap_diff Observed weighted mean difference (var1 - var2).

p_value Monte Carlo permutation p-value.

Examples

```
## Not run:
var1 <- rnorm(100)
var2 <- rnorm(100)
weights <- runif(100)
result <- shapley:::feature.test(var1, var2, weights)
result$mean_wmshap_diff
result$p_value

## End(Not run)
```

---

h2o.get_ids *Extract model IDs from H2O AutoML or Grid objects*

Description

Extracts model IDs from a "H2OAutoML" object (via the leaderboard) or from a "H2OGrid" object.

Usage

```
h2o.get_ids(h2oboard)
```

Arguments

h2oboard	An object inheriting from "H2OAutoML" or "H2OGrid".

Value

A character vector of trained model IDs.

Author(s)

E. F. Haghish

Examples

```
## Not run:
library(h2o)
h2o.init(ignore_config = TRUE, nthreads = 2, bind_to_localhost = FALSE, insecure = TRUE)
prostate_path <- system.file("extdata", "prostate.csv", package = "h2o")
prostate <- h2o.importFile(path = prostate_path, header = TRUE)
y <- "CAPSULE"
prostate[,y] <- as.factor(prostate[,y])  #convert to factor for classification
aml <- h2o.automl(y = y, training_frame = prostate, max_runtime_secs = 30)

# get the model IDs
ids <- h2o.get_ids(aml)

## End(Not run)
```

`normalize` *Normalize a vector based on specified minimum and maximum values*

Description

This function normalizes a vector based on specified minimum and maximum values. If the minimum and maximum values are not specified, the function will use the minimum and maximum values of the vector (ignoring missing values).

Usage

```
normalize(x, min = NULL, max = NULL)
```

Arguments

`x`	numeric vector
`min`	minimum value
`max`	maximum value

Value

A numeric vector of the same length as `x`

Author(s)

E. F. Haghish

Examples

```
## Not run:
# the function is not exported
normalize(c(0, 5, 10))
normalize(c(1, 1, 1))
normalize(c(NA, 2, 3))

## End(Not run)
```

---

shapley *Weighted Mean SHAP (WMSHAP) and Confidence Interval for Multiple Models (tuning grid, stacked ensemble, etc.)*

Description

Computes Weighted Mean SHAP ratios (WMSHAP) and confidence intervals to assess feature importance across a collection of models (e.g., an H2O grid/AutoML leaderboard or base-learners of an ensemble). Instead of reporting SHAP contributions for a single model, this function summarizes feature importance across multiple models and weights each model by a chosen performance metric. Currently, only models trained by the `h2o` machine learning platform, `autoEnsemble`, and the `HMDA` R packages are supported.

Usage

```
shapley(
  models,
  newdata,
  plot = TRUE,
  performance_metric = "r2",
  standardize_performance_metric = FALSE,
  performance_type = "xval",
  minimum_performance = 0,
  method = "mean",
  cutoff = 0.01,
  top_n_features = NULL,
  n_models = 10,
  sample_size = NULL
)
```

Arguments

`models`	An H2O AutoML object, H2O grid object, `autoEnsemble` object, or a character vector of H2O model IDs.
`newdata`	An `H2OFrame` (i.e., a `data.frame`) already uploaded to the `h2o` server. SHAP contributions are computed on this data.
`plot`	Logical. If `TRUE`, plots the WMSHAP summary (via `shapley.plot()`).

`performance_metric`
: Character. Performance metric used to weight models. Options are `"r2"` (regression), `"aucpr"`, `"auc"`, and `"f2"` (classification metrics).

`standardize_performance_metric`
: Logical. If `TRUE`, rescales model weights so the weights sum to the number of included models. The default is `FALSE`.

`performance_type`
: Character. Specify which performance metric performance estimate to use: `"train"` for training data, `"valid"` for validation, or `"xval"` for cross-validation (default).

`minimum_performance`
: Numeric. Specify the minimum performance metric for a model to be included in calculating WMSHAP. Models below this threshold receive zero weight and are excluded. The default is `0`. Specifying a minimum performance can be used to compute WMSHAP for a set of competitive models.

`method`
: Character. Specify the method for selecting important features based on their WMSHAP. The default is `"mean"`, which selects features whose WMSHAP exceeds the `cutoff`. The alternative is `"lowerCI"`, which selects features whose lower bound of confidence interval exceeds the `cutoff`.

`cutoff`
: Numeric. Cutoff applied by `method` for selecting important features.

`top_n_features`
: Integer or `NULL`. If not `NULL`, restricts SHAP computation to the top N features per model (reduces runtime). This also selects the top N features by WMSHAP in the returned `selectedFeatures`.

`n_models`
: Integer. Minimum number of models that must meet the performance threshold for WMSHAP and CI computation. Use `1` to compute summary SHAP for a single model. The default is 10.

`sample_size`
: Integer. Number of rows in `newdata` used for SHAP assessment. Defaults to all rows. Reducing this can speed up development runs.

Details

The function works as follows:

1. For each model, SHAP contributions are computed on `newdata`.
2. For each model, feature-level absolute SHAP contributions are aggregated and converted to a *ratio* (share of total absolute SHAP across features).
3. Models are weighted by a performance metric (e.g., `"r2"` for regression or `"auc"` / `"aucpr"` for classification).
4. The weighted mean SHAP ratio (WMSHAP) is computed for each feature, along with an confidence interval across models.

Value

An object of class `"shapley"` (a named list) containing:

ids Character vector of model IDs originally supplied or extracted.

included_models Character vector of model IDs included after filtering by performance.

ignored_models Data frame of excluded models and their performance.

weights Numeric vector of model weights (performance metrics) for included models.

results Data frame of row-level SHAP contributions merged across models.

summaryShaps Data frame of feature-level WMSHAP means and confidence intervals.

selectedFeatures Character vector of selected important features.

feature_importance List of per-feature absolute contribution summaries by model.

contributionPlot A ggplot-like object returned by `h2o.shap_summary_plot()` used for the WMSHAP ("wmshap") style plot.

plot A ggplot object (bar plot) if `plot = TRUE`, otherwise `NULL`.

Author(s)

E. F. Haghish

Examples

```
## Not run:
# load the required libraries for building the base-learners and the ensemble models
library(h2o)            #shapley supports h2o models
library(shapley)

# initiate the h2o server
h2o.init(ignore_config = TRUE, nthreads = 2, bind_to_localhost = FALSE, insecure = TRUE)

# upload data to h2o cloud
prostate_path <- system.file("extdata", "prostate.csv", package = "h2o")
prostate <- h2o.importFile(path = prostate_path, header = TRUE)

set.seed(10)

### H2O provides 2 types of grid search for tuning the models, which are
### AutoML and Grid. Below, I demonstrate how weighted mean shapley values
### can be computed for both types.

#########################################################
### PREPARE AutoML Grid (takes a couple of minutes)
#########################################################
# run AutoML to tune various models (GBM) for 60 seconds
y <- "CAPSULE"
prostate[,y] <- as.factor(prostate[,y])  #convert to factor for classification
aml <- h2o.automl(y = y, training_frame = prostate, max_runtime_secs = 120,
                  include_algos=c("GBM"),

               # this setting ensures the models are comparable for building a meta learner
                  seed = 2023, nfolds = 10,
                  keep_cross_validation_predictions = TRUE)

### call 'shapley' function to compute the weighted mean and weighted confidence intervals
### of SHAP values across all trained models.
### Note that the 'newdata' should be the testing dataset!
```

```
result <- shapley(models = aml, newdata = prostate, performance_metric = "aucpr", plot = TRUE)

#########################################################
### PREPARE H2O Grid (takes a couple of minutes)
#########################################################
# make sure equal number of "nfolds" is specified for different grids
grid <- h2o.grid(algorithm = "gbm", y = y, training_frame = prostate,
                 hyper_params = list(ntrees = seq(1,50,1)),
                 grid_id = "ensemble_grid",

              # this setting ensures the models are comparable for building a meta learner
                 seed = 2023, fold_assignment = "Modulo", nfolds = 10,
                 keep_cross_validation_predictions = TRUE)

result2 <- shapley(models = grid, newdata = prostate, performance_metric = "aucpr", plot = TRUE)

#########################################################
### PREPARE autoEnsemble STACKED ENSEMBLE MODEL
#########################################################

### get the models' IDs from the AutoML and grid searches.
### this is all that is needed before building the ensemble,
### i.e., to specify the model IDs that should be evaluated.
library(autoEnsemble)
ids    <- c(h2o.get_ids(aml), h2o.get_ids(grid))
autoSearch <- ensemble(models = ids, training_frame = prostate, strategy = "search")
result3 <- shapley(models = autoSearch, newdata = prostate,
                   performance_metric = "aucpr", plot = TRUE)



## End(Not run)
```

---

shapley.domain *Compute and plot weighted mean SHAP contributions at group level (factors or domains)*

Description

Aggregates SHAP contributions across user-defined domains (groups of features), computes weighted mean and an 95 returns a plot plus summary tables.

Usage

```
shapley.domain(
  shapley,
  domains,
  plot = TRUE,
  print = FALSE,
  colorcode = NULL,
```

```
  xlab = "Domains"
)
```

Arguments

shapley	Object of class "shapley", as returned by the 'shapley' function
domains	Named list of character vectors. Each element name is a domain name; each element value is a character vector of feature names assigned to that domain.
plot	Logical. If TRUE, a bar plot of domain WMSHAP contributions is created.
print	Logical. If TRUE, prints the domain WMSHAP summary table.
colorcode	Character vector for specifying the color names for each domain in the plot.
xlab	Character. Specify the ggplot 'xlab' label in the plot (default is "Domains")

Value

A list with:

domainSummary Data frame with WMSHAP domain contributions and CI.

domainRatio Data frame with per-model WMSHAP domain contribution ratios.

plot A ggplot object (or NULL if plotting not requested/implemented).

Author(s)

E. F. Haghish

Examples

```
## Not run:
# load the required libraries for building the base-learners and the ensemble models
library(h2o)            #shapley supports h2o models
library(shapley)

# initiate the h2o server
h2o.init(ignore_config = TRUE, nthreads = 2, bind_to_localhost = FALSE, insecure = TRUE)

# upload data to h2o cloud
prostate_path <- system.file("extdata", "prostate.csv", package = "h2o")
prostate <- h2o.importFile(path = prostate_path, header = TRUE)

### H2O provides 2 types of grid search for tuning the models, which are
### AutoML and Grid. Below, I demonstrate how weighted mean shapley values
### can be computed for both types.

set.seed(10)

#########################################################
### PREPARE AutoML Grid (takes a couple of minutes)
#########################################################
# run AutoML to tune various models (GBM) for 60 seconds
```

```
y <- "CAPSULE"
prostate[,y] <- as.factor(prostate[,y])  #convert to factor for classification
aml <- h2o.automl(y = y, training_frame = prostate, max_runtime_secs = 120,
                  include_algos=c("GBM"),

               # this setting ensures the models are comparable for building a meta learner
                  seed = 2023, nfolds = 10,
                  keep_cross_validation_predictions = TRUE)

### call 'shapley' function to compute the weighted mean and weighted confidence intervals
### of SHAP values across all trained models.
### Note that the 'newdata' should be the testing dataset!
result <- shapley(models = aml, newdata = prostate, plot = TRUE)

#########################################################
### PLOT THE WEIGHTED MEAN SHAP VALUES
#########################################################

shapley.plot(result, plot = "bar")

#########################################################
### DEFINE DOMAINS (GROUPS OF FEATURES OR FACTORS)
#########################################################
shapley.domain(shapley = result, plot = TRUE,
               domains = list(Demographic = c("RACE", "AGE"),
                              Cancer = c("VOL", "PSA", "GLEASON"),
                              Tests = c("DPROS", "DCAPS")),
                              print = TRUE)

## End(Not run)
```

`shapley.domain.test` *Weighted permutation test for difference between two domains*

Description

Computes domain-level contribution ratios (via `shapley.domain()`) and tests whether two domains differ using a weighted paired permutation test across models.

Usage

```
shapley.domain.test(shapley, domains, n = 2000)
```

Arguments

`shapley`	Object of class `"shapley"`, as returned by the 'shapley' function
`domains`	A named list of length 2. Each element is a character vector of feature names defining a domain; the two element names are the domain labels to be compared.
`n`	Integer, number of permutations (default 2000)

Value

A list with `mean_wmshap_diff` (observed weighted mean difference) and `p_value`.

Author(s)

E. F. Haghish

Examples

```
## Not run:
# load the required libraries for building the base-learners and the ensemble models
library(h2o)            #shapley supports h2o models
library(autoEnsemble)  #autoEnsemble models, particularly useful under severe class imbalance
library(shapley)

# initiate the h2o server
h2o.init(ignore_config = TRUE, nthreads = 2, bind_to_localhost = FALSE, insecure = TRUE)

# upload data to h2o cloud
prostate_path <- system.file("extdata", "prostate.csv", package = "h2o")
prostate <- h2o.importFile(path = prostate_path, header = TRUE)

### H2O provides 2 types of grid search for tuning the models, which are
### AutoML and Grid. Below, I demonstrate how weighted mean shapley values
### can be computed for both types.

set.seed(10)

#########################################################
### PREPARE AutoML Grid (takes a couple of minutes)
#########################################################
# run AutoML to tune various models (GBM) for 60 seconds
y <- "CAPSULE"
prostate[,y] <- as.factor(prostate[,y])  #convert to factor for classification
aml <- h2o.automl(y = y, training_frame = prostate, max_runtime_secs = 120,
                  include_algos=c("GBM"),

              # this setting ensures the models are comparable for building a meta learner
                  seed = 2023, nfolds = 10,
                  keep_cross_validation_predictions = TRUE)

### call 'shapley' function to compute the weighted mean and weighted confidence intervals
### of SHAP values across all trained models.
### Note that the 'newdata' should be the testing dataset!
result <- shapley(models = aml, newdata = prostate, plot = TRUE)

#########################################################
### Significance testing of contributions of two domains (or latent factors)
#########################################################
domains = list(Demographic = c("RACE", "AGE"),
               Cancer = c("VOL", "PSA", "GLEASON"))
shapley.domain.test(result, domains = domains, n=5000)
```

```
## End(Not run)
```

`shapley.feature.test` *Weighted permutation test for WMSHAP difference between two features*

Description

Performs a weighted paired permutation test to assess whether two features have different contributions (e.g., weighted mean SHAP, referred to as WMSHAP) across models in a `shapley` object.

Usage

```
shapley.feature.test(shapley, features, n = 2000)
```

Arguments

`shapley`	object of class `"shapley"`, as returned by the 'shapley' function
`features`	Character vector of length 2 giving the names of the two features to compare.
`n`	Integer. Number of permutations (default 2000).

Value

A list with `mean_wmshap_diff` (observed weighted mean difference) and `p_value`.

Author(s)

E. F. Haghish

Examples

```
## Not run:
# load the required libraries for building the base-learners and the ensemble models
library(h2o)            #shapley supports h2o models
library(autoEnsemble)  #autoEnsemble models, particularly useful under severe class imbalance
library(shapley)

# initiate the h2o server
h2o.init(ignore_config = TRUE, nthreads = 2, bind_to_localhost = FALSE, insecure = TRUE)

# upload data to h2o cloud
prostate_path <- system.file("extdata", "prostate.csv", package = "h2o")
prostate <- h2o.importFile(path = prostate_path, header = TRUE)

### H2O provides 2 types of grid search for tuning the models, which are
### AutoML and Grid. Below, I demonstrate how weighted mean shapley values
### can be computed for both types.
```

```
set.seed(10)

#######################################################
### PREPARE AutoML Grid (takes a couple of minutes)
#######################################################
# run AutoML to tune various models (GBM) for 60 seconds
y <- "CAPSULE"
prostate[,y] <- as.factor(prostate[,y])  #convert to factor for classification
aml <- h2o.automl(y = y, training_frame = prostate, max_runtime_secs = 120,
                  include_algos=c("GBM"),

              # this setting ensures the models are comparable for building a meta learner
                  seed = 2023, nfolds = 10,
                  keep_cross_validation_predictions = TRUE)

### call 'shapley' function to compute the weighted mean and weighted confidence intervals
### of SHAP values across all trained models.
### Note that the 'newdata' should be the testing dataset!
result <- shapley(models = aml, newdata = prostate, plot = TRUE)

#######################################################
### Significance testing of contributions of two features
#######################################################

shapley.feature.test(result, features = c("GLEASON", "PSA"), n = 5000)

## End(Not run)
```

`shapley.plot` *Plot weighted mean SHAP (WMSHAP) contributions*

Description

Visualizes WMSHAP summaries from a `shapley` object. Features can be selected using `method` and `method/cutoff`, `top_n_features`, or explicit `features` to specify feature selection method.

Usage

```
shapley.plot(
  shapley,
  plot = "bar",
  method = "mean",
  cutoff = 0.01,
  top_n_features = NULL,
  features = NULL,
  legendstyle = "continuous",
  scale_colour_gradient = NULL,
  labels = NULL
)
```

Arguments

shapley	object of class "shapley", as returned by the 'shapley' function
plot	Character. One of "bar" or "wmshap".
method	Character. One of "mean" or "lowerCI"; used by feature.selection() for feature selection when top_n_features or features are not set.
cutoff	Numeric cutoff for method selection.
top_n_features	Integer. If set, selects top N features by WMSHAP (overrides cutoff and method arguments).
features	Character vector, specifying the feature to be plotted (overrides cutoff and method arguments).
legendstyle	Character. For plot = "wmshap" only: "continuous" (default) or "discrete".
scale_colour_gradient	Optional character vector of length 3, specifying color names: c(low, mid, high). Used only when plot = "wmshap".
labels	Optional named character vector mapping feature names to display labels. To specify the labels, use the c function and for each feature, provide a label. For example, c(feature1 = label2, feature2 = label2, ...).

Value

A ggplot object

Author(s)

E. F. Haghish

Examples

```
## Not run:
# load the required libraries for building the base-learners and the ensemble models
library(h2o)            #shapley supports h2o models
library(shapley)

# initiate the h2o server
h2o.init(ignore_config = TRUE, nthreads = 2, bind_to_localhost = FALSE, insecure = TRUE)

# upload data to h2o cloud
prostate_path <- system.file("extdata", "prostate.csv", package = "h2o")
prostate <- h2o.importFile(path = prostate_path, header = TRUE)

### H2O provides 2 types of grid search for tuning the models, which are
### AutoML and Grid. Below, I demonstrate how weighted mean shapley values
### can be computed for both types.

set.seed(10)

#########################################################
### PREPARE AutoML Grid (takes a couple of minutes)
```

```
#########################################################
# run AutoML to tune various models (GBM) for 60 seconds
y <- "CAPSULE"
prostate[,y] <- as.factor(prostate[,y])  #convert to factor for classification
aml <- h2o.automl(y = y, training_frame = prostate, max_runtime_secs = 120,
                  include_algos=c("GBM"),

              # this setting ensures the models are comparable for building a meta learner
                  seed = 2023, nfolds = 10,
                  keep_cross_validation_predictions = TRUE)

### call 'shapley' function to compute the weighted mean and weighted confidence intervals
### of SHAP values across all trained models.
### Note that the 'newdata' should be the testing dataset!
result <- shapley(models = aml, newdata = prostate, plot = TRUE)

#########################################################
### PLOT THE WEIGHTED MEAN SHAP VALUES
#########################################################

shapley.plot(result, plot = "bar")
shapley.plot(result, plot = "wmshap")

## End(Not run)
```

`shapley.row.plot` *WMSHAP row-level plot for a single observation (participant or data row)*

Description

Computes and visualizes Weighted Mean SHAP contributions (WMSHAP) for a single row (subject/observation) across multiple models in a `shapley` object. For each feature, the function computes a weighted mean of row-level SHAP contributions across models using `shapley$weights` and reports an approximate 95 interval summarizing variability across models.

Usage

```
shapley.row.plot(
  shapley,
  row_index,
  top_n_features = NULL,
  features = NULL,
  nonzeroCI = FALSE,
  plot = TRUE,
  print = FALSE
)
```

Arguments

shapley	object of class "shapley", as returned by the 'shapley' function
row_index	Integer (length 1). The row/subject identifier to visualize. This is matched against the index column in shapley$results.
top_n_features	Integer. If specified, the top n features with the highest weighted SHAP values will be selected. This will be overrulled by the 'features' argument.
features	Optional character vector of feature names to plot. If NULL, all available features in shapley$results are used. Specifying the features argument will override the top_n_features argument.
nonzeroCI	Logical. If TRUE, it avoids ploting features that have a confidence interval crossing zero.
plot	Logical. If TRUE, prints the plot.
print	Logical. If TRUE, prints the computed summary table for the row.

Value

a list including the GGPLOT2 object and the data frame of WMSHAP summary values.

Author(s)

E. F. Haghish

Examples

```
## Not run:
# load the required libraries for building the base-learners and the ensemble models
library(h2o)            #shapley supports h2o models
library(shapley)

# initiate the h2o server
h2o.init(ignore_config = TRUE, nthreads = 2, bind_to_localhost = FALSE,
         insecure = TRUE)

# upload data to h2o cloud
prostate_path <- system.file("extdata", "prostate.csv", package = "h2o")
prostate <- h2o.importFile(path = prostate_path, header = TRUE)

set.seed(10)

### H2O provides 2 types of grid search for tuning the models, which are
### AutoML and Grid. Below, I demonstrate how weighted mean shapley values
### can be computed for both types.

#######################################################
### EXAMPLE 1: PREPARE AutoML Grid (takes a couple of minutes)
#######################################################
# run AutoML to tune various models (GBM) for 60 seconds
y <- "CAPSULE"
prostate[,y] <- as.factor(prostate[,y])  #convert to factor for classification
```

```
aml <- h2o.automl(y = y, training_frame = prostate, max_runtime_secs = 120,
                  include_algos=c("GBM"),

                  seed = 2023, nfolds = 10,
                  keep_cross_validation_predictions = TRUE)

### call 'shapley' function to compute the weighted mean and weighted confidence intervals
### of SHAP values across all trained models.
### Note that the 'newdata' should be the testing dataset!
result <- shapley(models = aml, newdata = prostate,
                   performance_metric = "aucpr", plot = TRUE)

shapley.row.plot(result, row_index = 11)

#########################################################
### EXAMPLE 2: PREPARE H2O Grid (takes a couple of minutes)
#########################################################
# make sure equal number of "nfolds" is specified for different grids
grid <- h2o.grid(algorithm = "gbm", y = y, training_frame = prostate,
                  hyper_params = list(ntrees = seq(1,50,1)),
                  grid_id = "ensemble_grid",

               # this setting ensures the models are comparable for building a meta learner
                  seed = 2023, fold_assignment = "Modulo", nfolds = 10,
                  keep_cross_validation_predictions = TRUE)

result2 <- shapley(models = grid, newdata = prostate,
                    performance_metric = "aucpr", plot = TRUE)

shapley.row.plot(result2, row_index = 9)
shapley.row.plot(result2, row_index = 9, nonzeroCI = TRUE)
shapley.row.plot(result2, row_index = 9, top_n_features = 10)

#########################################################
### EXAMPLE 3: PREPARE autoEnsemble STACKED ENSEMBLE MODEL
#########################################################

### get the models' IDs from the AutoML and grid searches.
### this is all that is needed before building the ensemble,
### i.e., to specify the model IDs that should be evaluated.
library(autoEnsemble)
ids    <- c(h2o.get_ids(aml), h2o.get_ids(grid))
autoSearch <- ensemble(models = ids, training_frame = prostate, strategy = "search")
result3 <- shapley(models = autoSearch, newdata = prostate,
                    performance_metric = "aucpr", plot = TRUE)

#plot all important features
shapley.row.plot(result3, row_index = 13)

#plot only the given features
shapPlot <- shapley.row.plot(result3, row_index = 13, features = c("PSA", "AGE"))

# inspect the computed data for the row 13
```

```
ptint(shapPlot$summary)

## End(Not run)
```

---

shapley.table *Create WMSHAP Summary Table Based on the Given Criterion*

Description

#' Generates a summary table of weighted mean SHAP ratios (WMSHAP) and confidence intervals for each feature based on a weighted SHAP analysis. The function filters the SHAP summary table (from a shapley object) by selecting features that meet or exceed a specified cutoff using a selection method (default "mean").

The output is sorted by WMSHAP and formatted as either a markdown table (via **pander**) or a data frame.

Usage

```
shapley.table(
  shapley,
  method = "mean",
  cutoff = 0.01,
  round = 3,
  exclude_features = NULL,
  dict = NULL,
  markdown.table = TRUE,
  split.tables = 120,
  split.cells = 50
)
```

Arguments

shapley	A shapley object returned by shapley that contains a data frame summaryShaps.
method	Character. The column name in summaryShaps used for feature selection. Default is "mean", which selects important features which have weighted mean shap ratio (WMSHAP) higher than the specified cutoff. Other alternative is "lowerCI", which selects features which their lower bound of confidence interval is higher than the cutoff.
cutoff	Numeric. The threshold cutoff for the selection method; only features with a value in the method column greater than or equal to this value are retained. Default is 0.01.
round	Integer. The number of decimal places to round the SHAP mean and confidence interval values. Default is 3.
exclude_features	Character vector. A vector of feature names to be excluded from the summary table. Default is NULL.

dict	A data frame containing at least two columns named "name" and "description". If provided, the function uses this dictionary to add human-readable feature descriptions. Default is NULL.
markdown.table	Logical. If TRUE, the output is formatted as a markdown table using the **pander** package; otherwise, a data frame is returned. Default is TRUE.
split.tables	Integer. Controls table splitting in pander(). Default is 120.
split.cells	Integer. Controls cell splitting in pander(). Default is 50.

Value

If markdown.table = TRUE, returns a markdown table (invisibly) showing two columns: "Description" and "WMSHAP". If markdown.table = FALSE, returns a data frame with these columns.

Author(s)

E. F. Haghish

Examples

```
## Not run:
# load the required libraries for building the base-learners and the ensemble models
library(h2o)            #shapley supports h2o models
library(shapley)

# initiate the h2o server
h2o.init(ignore_config = TRUE, nthreads = 2, bind_to_localhost = FALSE, insecure = TRUE)

# upload data to h2o cloud
prostate_path <- system.file("extdata", "prostate.csv", package = "h2o")
prostate <- h2o.importFile(path = prostate_path, header = TRUE)

set.seed(10)

### H2O provides 2 types of grid search for tuning the models, which are
### AutoML and Grid. Below, I demonstrate how weighted mean shapley values
### can be computed for both types.

#########################################################
### PREPARE AutoML Grid (takes a couple of minutes)
#########################################################
# run AutoML to tune various models (GBM) for 60 seconds
y <- "CAPSULE"
prostate[,y] <- as.factor(prostate[,y])  #convert to factor for classification
aml <- h2o.automl(y = y, training_frame = prostate, max_runtime_secs = 120,
                  include_algos=c("GBM"),

              # this setting ensures the models are comparable for building a meta learner
                  seed = 2023, nfolds = 10,
                  keep_cross_validation_predictions = TRUE)

### call 'shapley' function to compute the weighted mean and weighted confidence intervals
```

```
### of SHAP values across all trained models.
### Note that the 'newdata' should be the testing dataset!
result <- shapley(models = aml, newdata = prostate, performance_metric = "aucpr", plot = TRUE)

#######################################################
### PREPARE H2O Grid (takes a couple of minutes)
#######################################################
# make sure equal number of "nfolds" is specified for different grids
grid <- h2o.grid(algorithm = "gbm", y = y, training_frame = prostate,
                 hyper_params = list(ntrees = seq(1,50,1)),
                 grid_id = "ensemble_grid",

              # this setting ensures the models are comparable for building a meta learner
                 seed = 2023, fold_assignment = "Modulo", nfolds = 10,
                 keep_cross_validation_predictions = TRUE)

result2 <- shapley(models = grid, newdata = prostate, performance_metric = "aucpr", plot = TRUE)

# get the output as a Markdown table:
md_table <- shapley.table(shapley = result2,
                          method = "mean",
                          cutoff = 0.01,
                          round = 3,
                          markdown.table = TRUE)
head(md_table)

## End(Not run)
```

`shapley.top` *Flag and rank features by WMSHAP cutoffs*

Description

This function applies different criteria simultaniously to identify the most important features in a model. The criteria include: 1) minimum limit of lower weighted confidence intervals of SHAP values relative to the feature with highest SHAP value. 2) minimum limit of percentage of weighted mean SHAP values relative to over all SHAP values of all features. These are specified with two different cutoff values.

Usage

```
shapley.top(shapley, mean = 0.01, lowerCI = 0.01)
```

Arguments

`shapley`	object of class 'shapley', as returned by the 'shapley' function

mean	Numeric. specifying the cutoff of weighted mean SHAP ratio (WMSHAP). The default is 0.01. Lower values will be more generous in defining "importance", while higher values are more restrictive. However, these default values are not generalizable to all situations and algorithms.
lowerCI	numeric. Specifying the limit of lower bound of 95% WMSHAP The default is 0.01. Lower values will be more generous in defining "importance", while higher values are more restrictive. However, these default values are not generalizable to all situations and algorithms.

Value

data.frame of selected features

Author(s)

E. F. Haghish

Examples

```
## Not run:
# load the required libraries for building the base-learners and the ensemble models
library(h2o)            #shapley supports h2o models
library(shapley)

# initiate the h2o server
h2o.init(ignore_config = TRUE, nthreads = 2, bind_to_localhost = FALSE, insecure = TRUE)

# upload data to h2o cloud
prostate_path <- system.file("extdata", "prostate.csv", package = "h2o")
prostate <- h2o.importFile(path = prostate_path, header = TRUE)

### H2O provides 2 types of grid search for tuning the models, which are
### AutoML and Grid. Below, I demonstrate how weighted mean shapley values
### can be computed for both types.

set.seed(10)

#########################################################
### PREPARE AutoML Grid (takes a couple of minutes)
#########################################################
# run AutoML to tune various models (GBM) for 60 seconds
y <- "CAPSULE"
prostate[,y] <- as.factor(prostate[,y])  #convert to factor for classification
aml <- h2o.automl(y = y, training_frame = prostate, max_runtime_secs = 120,
                  include_algos=c("GBM"),

              # this setting ensures the models are comparable for building a meta learner
                  seed = 2023, nfolds = 10,
                  keep_cross_validation_predictions = TRUE)

### call 'shapley' function to compute the weighted mean and weighted confidence intervals
### of SHAP values across all trained models.
```

```
### Note that the 'newdata' should be the testing dataset!
result <- shapley(models = aml, newdata = prostate, plot = TRUE)

#######################################################
### Select top features
#######################################################
shapley.top(result, mean = 0.005, lowerCI = 0.01)

## End(Not run)
```

Index